



Europäisches  
Patentamt

European  
Patent Office

Office européen  
des brevets

#8

Bescheinigung

Certificate

Attestation



Die angehefteten Unterla-  
gen stimmen mit der  
ursprünglich eingereichten  
Fassung der auf dem näch-  
sten Blatt bezeichneten  
europäischen Patentanmel-  
dung überein.

The attached documents  
are exact copies of the  
European patent application  
described on the following  
page, as originally filed.

Les documents fixés à  
cette attestation sont  
conformes à la version  
initialement déposée de  
la demande de brevet  
européen spécifiée à la  
page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

99480050.6

CERTIFIED COPY OF  
PRIORITY DOCUMENT

Der Präsident des Europäischen Patentamts;  
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets  
p.o.

C. PASTUREL

DEN HAAG, DEN

Best Available Copy

**THIS PAGE BLANK (USPTO)**



Europäisches  
Patentamt

European  
Patent Office

Office européen  
des brevets

**Blatt 2 der Bescheinigung**  
**Sheet 2 of the certificate**  
**Page 2 de l'attestation**

Anmeldung Nr.:  
Application no.: 99480050.6  
Demande n°:

Anmeldetag:  
Date of filing: 01/07/99  
Date de dépôt:

Anmelder:  
Applicant(s):  
Demandeur(s):  
INTERNATIONAL BUSINESS MACHINES CORPORATION  
Armonk, NY 10504  
UNITED STATES OF AMERICA

Bezeichnung der Erfindung:  
Title of the invention:  
Titre de l'invention:

Hardware device for processing the tasks of an algorithm in parallel

In Anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

Staat:  
State:  
Pays:

Tag:  
Date:  
Date:

Aktenzeichen:  
File no.  
Numéro de dépôt:

Internationale Patentklassifikation:  
International Patent classification:  
Classification internationale des brevets:

/

Am Anmeldetag benannte Vertragsstaaten:  
Contracting states designated at date of filing: AT/BE/CH/CY/DE/DK/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/PT/SE  
Etats contractants désignés lors du dépôt:

Bemerkungen:  
Remarks:  
Remarques:

**THIS PAGE BLANK (USPTO)**

## HARDWARE DEVICE FOR PROCESSING THE TASKS OF AN ALGORITHM IN PARALLEL

### Technical field

5 The invention relates to the processing of algorithms used in particular in the search engines of a large data communication network such as Internet, and relates particularly to a hardware device for processing the tasks of an algorithm in parallel.

### Background

15 The World Wide Web (WWW) provides a large source of information. Compared with traditional databases, Web information is dynamic and structured with hyperlinks. Also, it can be represented in different forms and is globally shared over multiple sites and platforms. Hence, querying over the WWW is significantly different from querying the data from traditional databases, e.g. relational databases, which are structured, centralized and static. It can cope with tens of information sources; but it is ineffective for thousands.

20 Most Web documents are text-oriented. Many relevant information is usually embedded in the text and could not be explicitly or easily specified in a user query. To facilitate Web searching, many search engines and similar programs have been developed. Most of these programs are database based meaning that the  
25 system maintains a database, a user searches the web by specifying a set of keywords and formulating a query to the database. Web search aids are variously referred to as catalogs, directories, indexes, search engines, or Web databases.

30 A search engine is a Web site on the Internet which someone may use to find desired Web pages and sites. A search engine will

generally return the results of a user's search ranked by relevancy.

A competent Web search engine must include the fundamental search facilities that Internet users are familiar with, which include Boolean logic, phrase searching, truncation, and limiting facilities (e.g. limit by field). Most of the services try more or less to index the full-text of the original documents, which allows the user to find quite specialised information. Most services use best match retrieval systems, someones use a Boolean system only.

Web search engines execute algorithms having internal processes which are repetitive tasks with independent entry data. A classical step by step processing of all processes and decisions on one entry data before processing the next entry data has the drawback to take much time to process all the data. Thus, it is the case to perform the search of a pattern within each file of a disk. The main repetitive processes to perform are : load file, open file, scan each word and compare for matching with a pattern, append the result in a temporary file, close file.

One way to improve the performance, and in particular to improve the search response time, is to achieve parallel processing by parallelizing the search mechanism in the database or index table. Such software parallelization will be more optimized but is nevertheless limited insofar as the software processing, even if parallelized, requires a minimum of time which cannot be reduced.

#### **Summary of the invention**

Accordingly, the object of the invention is to provide a hardware assist device able to run a set of repetitive processes using local pipelining for each task, and maintaining

a relationship between the parent task and the child task for each occurrence in the pipeline.

Another object of the invention is to provide a hardware device for processing the tasks of a search algorithm in parallel wherein each specific task of the search is made by a dedicated processor.

The invention relates therefore to a hardware device for processing the tasks of an algorithm of the type comprising a number of processes the execution of some of which depending on binary decisions, the device comprising a plurality of task units which are each associated with a task defined as being either one process or one decision or one process together with the following decision, and a task interconnection logic block connected to each task unit for communicating actions from a source task unit to a destination task unit, each task unit including a processor for processing the steps of the associated task when the received action requests such a processing and a status manager for handling the actions coming from other task units and building the actions to be sent to other task units

#### **Brief description of the drawings**

The above and other objects, features and advantages of the invention will be better understood by reading the following more particular description of the invention in conjunction with the accompanying drawings wherein :

Fig. 1 represents an example of algorithm composed of three processes and three decisions.

Fig. 2 represents the algorithm illustrated in Fig. 1 which has been structured into several tasks to be executed by the hardware device according to the invention.

Fig. 3 is a block-diagram representing the hardware device according to the invention.

Fig. 4 is a representation of the configuration register used to control each task executed by the hardware device of Fig. 3.

5 Fig. 5A and 5B are tables representing respectively the actions to be executed by each task of the algorithm illustrated in Fig. 1 in function of the possible activation sources for an instance and the following instance.

10 Fig. 6 is a block-diagram representing the connection between the task inteconnection logic block of the hardware device of Fig. 3 and the different tasks of the algorithm.

### Detailed description of the invention

15 The simple algorithm illustrated in Fig. 1 as an example includes three processes  $P_1$ ,  $P_2$  and  $P_3$  and two decisions  $D_1$  and  $D_2$ . Depending on each decision, different functions corresponding to the different paths in the algorithm may be run. The first function is represented by the algorithm flow when decision  $D_1$  is «yes», that is when processes  $P_1$  and  $P_2$  are to be executed. The second function is represented by the algorithm flow when decision  $D_1$  is «no» and decision  $D_2$  is «yes», that is when processes  $P_1$  and  $P_3$  are to be executed. 20 Finally, the third function is represented by the algorithm flow when decision  $D_1$  is «no» and decision  $D_2$  is also «no», that is when only process  $P_1$  is to be executed. In the latter case, the algorithm flow loops back to the entry point and the same functions may be executed again. Thus, during the first 25 algorithm flow, process  $P_1$  is started while the execution of process  $P_1$  is started again when decision  $D_2$  is «no». The second execution of  $P_1$  starts only when the first execution of  $P_1$  has been completed and when decision  $D_1$  and  $D_2$  have been completed. 30 Therefore, there is no overlap possible in a simple step by step processing of the algorithm.

Though the algorithm represented in Fig. 1 is very simple, all the algorithms are classically run in the same way.



All the events (processes or decisions) of the algorithm flow have to be executed step by step although they are run repetitively with new entry data. The proposed invention allows to run separately the various processes and decisions to speed up the processing of the algorithm especially when there is no prior data required on some steps. The main idea to achieve this is to have one processor assigned to a «task» including a process, a decision or a combination of processes and decisions which will run all the repetitive instances of this task and will be linked to the execution result of the other task processors using a more detailed link information than the simple conventional link enabling the downstream tasks to be activated.

Using the principles of the invention, the algorithm of Fig. 1 can be divided into tasks as illustrated in Fig. 2. Four tasks are thus implemented.

- Task 1 ( $T_1$ ) includes process  $P_2$  (no decision)
- Task 2 ( $T_2$ ) includes process  $P_3$  (no decision)
- Task 3 ( $T_3$ ) includes the sequential combination of process  $P_1$  and decision  $D_1$
- Task 4 ( $T_4$ ) includes only decision  $D_2$  (no process)

According to the invention, each task is repetitively performed by one processor allocated to this task. Therefore, four processors will be required to run the example algorithm of Fig. 1 and Fig. 2.

The hardware device according to the invention illustrated in Fig. 3 comprises as many task units 10, 12, 14 as the number of tasks included in the algorithm (Task  $_1$ , Task $_2$  .... Task $_n$ ). The interconnection between the tasks is performed by the intermediary of a Task interconnection logic block 16 as explained hereafter.

Each task unit like task unit 10 includes a processor 18 in charge of processing the sequential steps of the process, the decision or the combination of the process and the decision generally incorporated in the corresponding task. Actions received from other task units or sent to other task units by means of Task interconnection logic block 16 are managed by status manager 20 which is preferably a state machine. Status manager 20 is connected to processor 18 by two lines, an input line to processor 18 for starting (S) the task execution and the output line from the processor which is activated when the task is completed (C).

Status manager 20 has essentially two functions : input and output. The input function handles incoming commands from other tasks and the output function builds commands to be sent to other tasks. To perform these functions in conjunction with processor 18, several control/data registers 22, 24, 26 are used. Each control/data register corresponds, for this task, to an instance of the algorithm flow. The number of instances which can be run at the same time depends upon the pipeline capability of processor 18. Generally, it is necessary to have three control/data registers corresponding to instances  $m$ ,  $m+1$ ,  $m+2$ .

Each control/data register 22, 24 or 26 contains a control field and a data field. The control field is composed of three bits controlled by processor 18, a validation bit  $V$ , a completion bit  $C$  and a bit  $L/R$  indicating whether the output is Left of Right when the task includes a decision.

The data field of a control/data register contains data which are loaded by status manager 20 after receiving an action to be performed from another task and before starting the task execution by sending the start command to task processor 18. These data may be used by processor 18. When the latter has

completed the task execution, it may replace the data contained in the control/data register by other data. This data will then be sent to the destination task in the command word and used as an input field by the destination task processor. However, it must be noted that, in case of independent tasks, the data are not modified in the control/data register.

When the task execution has been completed by processor 18, this one sets to 1 the bit C of the control field of the control/data register and a signal C may be sent to status manager 20. Therefore, either status manager is activated by the input signal C from task processor 18, or there is a polling or an interrupt mechanism which enables the status manager to be informed of the setting of bit C to 1.

The commands which may be received from another task by status manager 20 are START, KILL or VALID. As already mentioned, the START command is used to activate task processor 18. The KILL command means that a task is no longer of interest since the taken decision is opposite to this task. Thus, a task which is the left path of a decision may be killed if the decision is to take the right path. When it receives a KILL command, status manager 20 clears the control/data register corresponding to the instance being considered as each command has as a parameter the instance value called level. Conversely to the KILL command, the VALID command confirms that the considered task corresponds to the taken decision. In such a case, the bit V of the corresponding control/data register is set to 1 by status manager 20.

The output function of status manager 20 is to build commands based on the contents of two configuration registers, CONFIG.L 28 and CONFIG.R 30 and also on the contents of the involved control/data register. The contents of CONFIG.L register which is selected when bit L/R set to 1 are given in Fig. 4. Note that the CONFIG.R register which is selected when bit L/R is

set to 0 has exactly the same structure as CONFIG.L register. Note that the CONFIG.L and CONFIG.R registers are loaded at the beginning of the algorithm processing and remain unchanged insofar as they contain data fields depending only on the algorithm structure.

As illustrated in Fig. 4, CONFIG.L register contains a first block C selected when bit C is set to 1 and a second block V selected when bit V is set to 1. Each block C or V is used for two actions. For each action the register contains the three following fields wherein  $X = C$  or  $V$  and  $n = 1$  or  $2$ .

- Task  $X_n$  indicates which task should be activated
- $A_{Xn}$  indicates which action is to be performed. For example 00 = kill, 01 = start, 10 = valid and 11 = valid + start.
- $L_{Xn}$  indicates the level of task (the instance) corresponding to Task  $X_n$ . For example, 00 = current level - 1, 01 = current level, 10 = current level + 1, 11 = current level + 2.

As an example enabling the invention to be understood, the algorithm illustrated in Fig. 2 can be considered. There are four tasks  $T_1$ ,  $T_2$ ,  $T_3$  and  $T_4$  which can be executed, but there are six activation sources since Task 3 and Task 4 have each two outputs. Furthermore, a task acting as a source task can activate a destination task in the same level or in the following level. Fig. 5A and Fig. 5B represent tables wherein the activation sources are associated with the columns whereas the tasks to be activated are associated with the rows. Fig. 5A corresponds to the activation of the tasks in a same level whereas Fig. 5B corresponds to the activation of the tasks in level  $m+1$  by activation sources in level  $m$ . It should be noted that since only two levels are represented, this means that there is no relationship between the processes of the algorithm on more than two consecutive levels.

In the tables illustrated in Fig. 5A and 5B, only the cases corresponding to an action from an activation source to a task are filled with a letter. Letter S means «Start», V means «Validate» and K means «Kill». It must be noted that it is possible that a same source has an action on two tasks. Thus, T<sub>3</sub>R kills Task 1, and starts and validates Task T<sub>4</sub>.

As already mentioned, status manager 20 (Fig. 2) uses the control bits which have been previously loaded in CONFIG.L and CONFIG.R registers associated with the task. Thus, if we consider Task3 which generates two activation sources, the CONFIG.L and CONFIG.R registers have the following contents :

#### CONFIG.L

##### 1. Bloc C

- Action 1    Task C<sub>1</sub> = Task 3  
                  AC<sub>1</sub>        = start  
                  LC<sub>1</sub>        = current level + 1
- Action 2    none

##### 2. Bloc V

- Action 1    Task V<sub>1</sub> = Task 1  
                  AV<sub>1</sub>        = valid  
                  LV<sub>1</sub>        = current level
- Action 2    none

#### CONFIG.R

##### 1. Bloc C

- Action 1    Task C<sub>1</sub> = Task 3  
                  AC<sub>1</sub>        = start  
                  LC<sub>1</sub>        = current level + 1
- Action 2    none

##### 2. Bloc V

- Action 1    Task V<sub>1</sub> = Task 1

AV<sub>1</sub> = kill  
LV<sub>1</sub> = current level  
- Action 2 Task V<sub>2</sub> = Task 4  
AV<sub>2</sub> = valid + start  
LV<sub>2</sub> = current level

5

The Task interconnection logic block 16 is represented in Fig. 6. Each task such as Task1, Task2, Task3, ... Taskn is an input to Task interconnection logic block 16 but is also an output to this block. Each input action or command could be of the same type as each one of the output actions such as KILL, START or VALID. Using the CONFIG.L and CONFIG.R registers where an action is represented by three control fields Task Xn, Axn and Lxn, an action word may use this control fields in addition to the corresponding data (see Fig.4) to transmit the action to the destination task.

10

15

In the preferred embodiment illustrated in Fig. 6, the action word containing the control bits of CONFIG.L or CONFIG.R registers and data is input to a three-state driver 40, 42, 44 or 46 where the Task Xn field is decoded in order to select on which bus this action word should be put. This word, or the remaining bits insofar as the Task Xn field is no longer used, are then decoded by the appropriate task to perform the requesting action.

20

As illustrated in Fig. 6, there are as many buses as the number of tasks. These buses are three-state so that all inactive inputs have no influence in the bus value. Only the valid one forced by the corresponding driver takes the bus for its command. The width of the bus depends on the size of the action word. In the preferred embodiment the bus size is equal to word size. If there is a problem in the size of the bus, it is well known how to split the word into several blocks appended when sent on a smaller bus. The only drawback of this split will be an increased transmission latency as it will need several clock

25

30

times to transmit a command or action from one output task to an input task. At least, the TaskXn should be available in the first block of the split word to be decoded correctly.

Each task can then put all the actions on the various buses. As  
5 long as there is no capability to have an action simultaneously put on the same bus by two tasks, there is no arbitration required. This is the case for most of the algorithms. Otherwise, an arbitration mechanism may be added on the control  
10 of each three-state driver to identify 2 simultaneous requests for the same destination. A very simple contention mechanism will for example give the priority on the destination bus to the lower source task.

**THIS PAGE BLANK (USPTO)**



## CLAIMS

1. Hardware device for processing the tasks of an algorithm of the type comprising a number of processes the execution of some of which depending on binary decisions ; said device comprising a plurality of task units (10, 12, 14) which are each associated with a task defined as being either one of said processes or one of said decisions or one of said processes together with the following decision, and a task interconnection logic block (16) connected to each task unit for communicating actions from a source task unit to a destination task unit, each task unit including a processor (18) for processing the steps of the associated task when the received action requests such a processing and a status manager (20) for handling the actions coming from other task units and building the actions to be sent to other task units
2. Hardware device according to claim 1, wherein said actions communicated from a source task unit to a destination task unit are START used to activate the processor (18) of said destination task unit, KILL used to cancel the task associated with said destination task unit and VALID used to confirm that task associated with said destination task unit corresponds to a decision included in said task.
3. Hardware device according to claim 2, wherein said status manager (20) activates said processor (18) for processing the steps of the task associated with said destination task unit when the action received from a source task unit is START.
4. Hardware device according to claim 3, wherein said status manager (20) is a state machine.

5. Hardware device according to any one of claims 2 to 4, wherein each of said task units (10, 12, 14) further comprises a plurality of control/data registers (22, 24, 26) each corresponding, for the task associated with said task unit, to an instance of the algorithm flow, each one of said control/data registers comprising a control field composed of a completion bit (C) set to 1 when the associated task is completed, a validation bit (V) set to 1 when the associated task is validated and a L/R bit indicating that the output in the algorithm flow is left or right when said task includes a decision.
6. Hardware device according to claim 5, wherein each of said control/data registers (22, 24, 26) includes a data field which is loaded if necessary by said status manager (20) activated by an action received from a source task unit, said processor (18) using these data for executing the task associated with said task unit and replacing them if necessary.
7. Hardware device according to claim 6, wherein said completion bit (C) is sent by said processor (18) to said status manager (20) after completion of the task execution.
8. Hardware device according to claim 5, 6 or 7, wherein said control/data register (22, 24 or 26) corresponding to a specific instance is cleared by said status manager (20) when this one receives an action KILL for the task associated with said task unit and for said specific instance.
9. Hardware device according to any one of claims 5 to 8, wherein each one of said task units (10, 12, 14) further comprises two configuration registers CONFIG.L (28) and CONFIG.R (30) which are respectively selected by the binary value of said bit L/R of the control/data register (22, 24,

26) of the instance being considered, the contents of said configuration registers being loaded at the beginning of the algorithm processing for defining the task to be activated, the action to be performed and the instance to be considered .

5

10. Hardware device according to any one of claims 1 to 9, wherein said task interconnection logic block (16) is composed of three-state drivers (40, 42, 44) each one of said drivers being associated with one of said tasks as input task and a number of buses equal to the number of said tasks as output tasks, one of said buses being selected by the driver corresponding to an input task after decoding an action word by said driver.

10

11. Hardware device according to claim 10 and claims 5 and 9, wherein said action word contains the control bits of said configuration registers (28, 30) and data from said control/data registers of said task unit (10, 12 or 14) associated with the input task.

15

**THIS PAGE BLANK (USPTO)**

FR 9 99 027  
Benayoun et al  
1/4

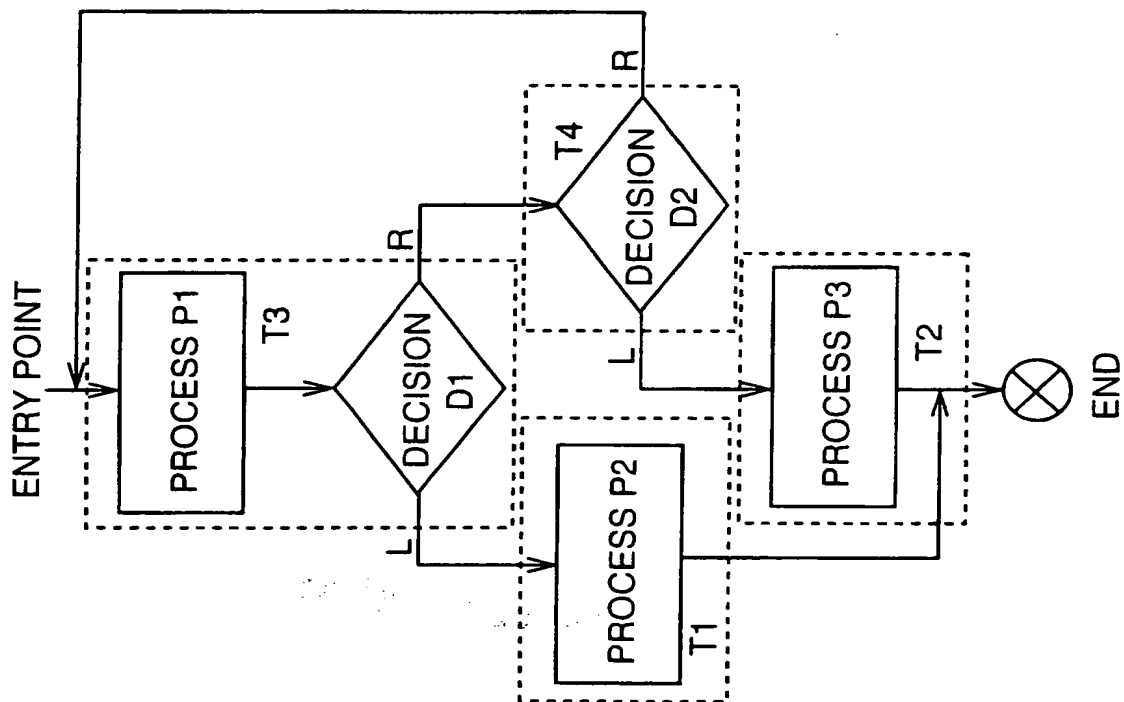


FIG. 2

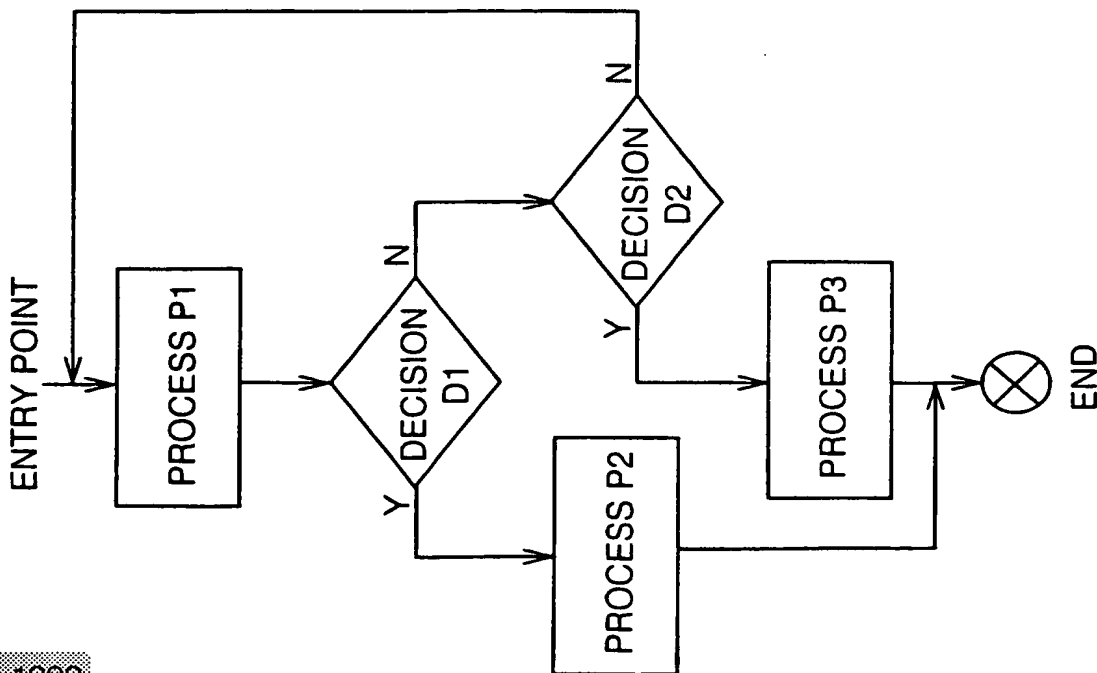


FIG. 1

FR 9 99 027  
Benayoun et al  
2/4

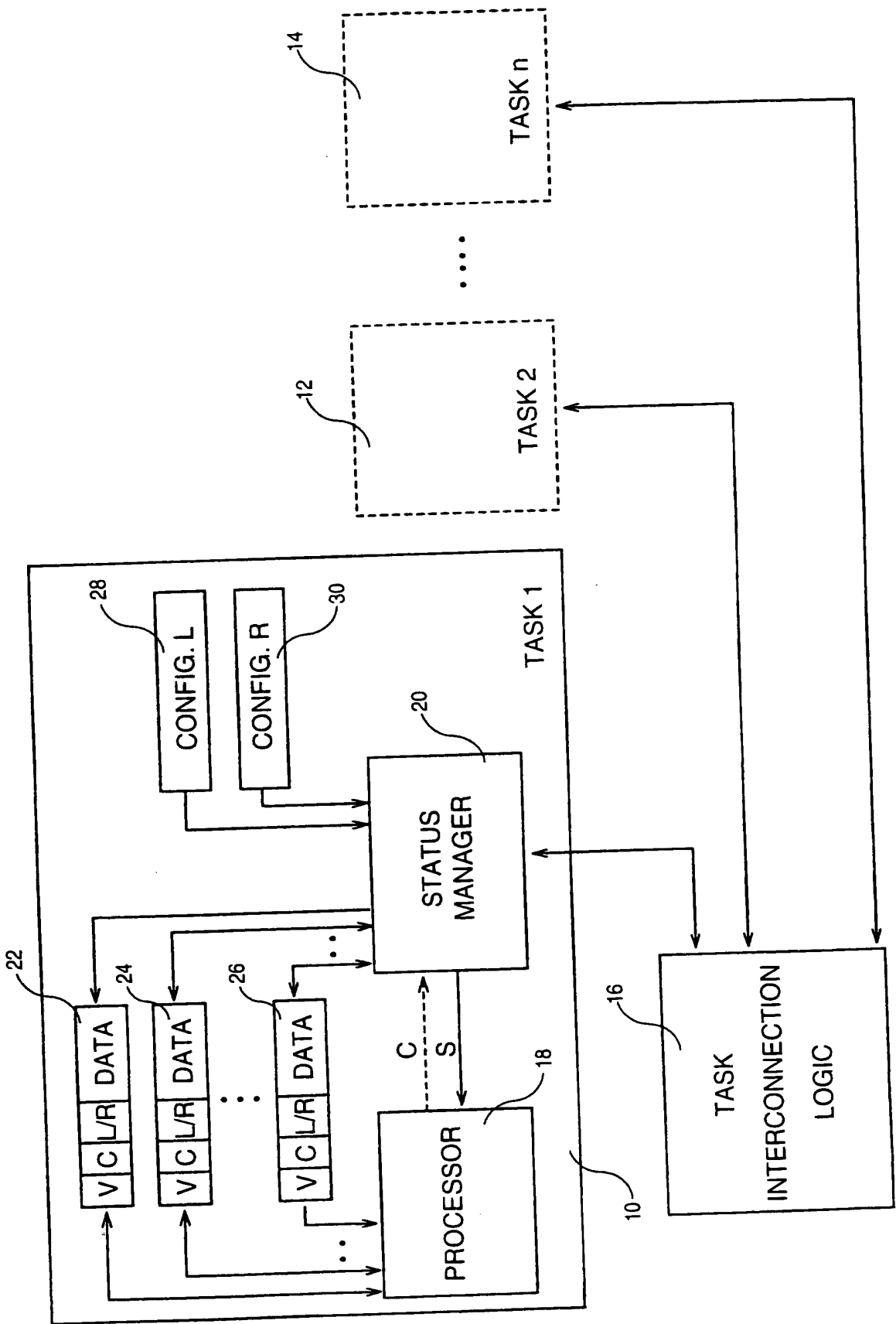


FIG. 3

FR 9 99 027  
Benayoun et al  
3/4

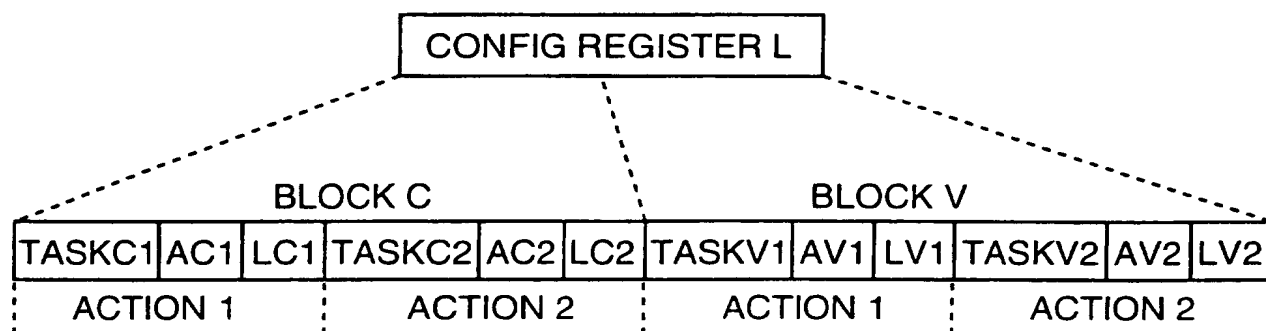


FIG. 4

LEVEL m to LEVEL m	T1	T2	T3L	T3R	T4L	T4R
T1			V	K		
T2					V	K
T3						
T4				S,V		

FIG. 5A

LEVEL m to LEVEL m+1	T1	T2	T3L	T3R	T4L	T4R
T1	S					
T2		S				
T3			S	S		V
T4						

FIG. 5B

FR 9 99 027  
Benayoun et al  
4/4

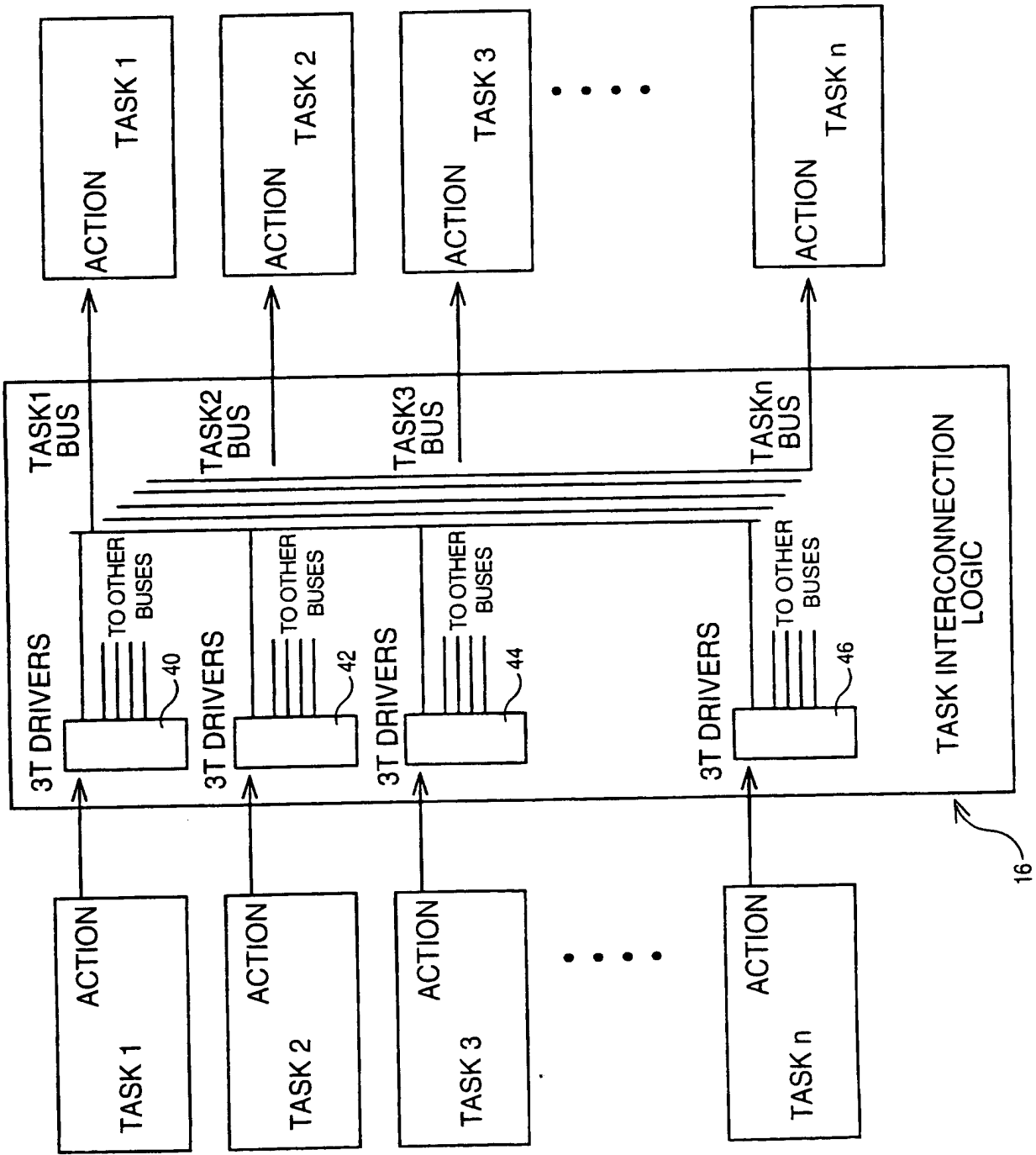


FIG. 6



# HARDWARE DEVICE FOR PROCESSING THE TASKS OF AN ALGORITHM IN PARALLEL

## Abstract

Hardware device for processing the tasks of an algorithm of the  
5 type comprising a number of processes the execution of some of  
which depending on binary decisions, the device comprising a  
plurality of task units (10, 12, 14) which are each associated  
with a task defined as being either one process or one decision  
or one processe together with the following decision, and a  
10 task interconnection logic block (16) connected to each task  
unit for communicating actions from a source task unit to a  
destination task unit, each task unit including a processor  
(18) for processing the steps of the associated task when the  
received action requests such a processing and a status manager  
15 (20) for handling the actions coming from other task units and  
building the actions to be sent to other task units

FIG. 3

**THIS PAGE BLANK (USPTO)**

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☒ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**

**THIS PAGE BLANK (USPTO)**